Resistance Project Report

Jasper Paterson 22736341

Jordan Hartley 22713786

1.0 Introduction

As humans learn and play games, they tend to develop strategies based on what has worked in the past. Gameplay decisions can also stem from social interactions between players, which can be both an advantage and a hinderance to winning the game. Don Eskridge's *The Resistance* is a hidden role board game that is no exception to this. One of the most important aspects of the game is the social interactions between players, as they both deceive and learn to trust each other. *The Resistance* is an interesting game for artificial intelligence (AI) to tackle, given that computers will play purely based on game logic and observable game actions rather than being able to leverage or be influenced by social interactions.

Over the course of the project, we have gained experience with *The Resistance* through playing social games and have found generally effective strategies. We have also developed an idea of the general flow of the game from the perspective of both a spy and a resistance member. The objective of this paper is to attempt to create an AI-optimised strategy that is superior to the way we would usually play in a game as humans. To do this we have developed two AI agents that can play games of *The Resistance*. We will allow one agent to modify and attempt to optimise its own strategy and behaviours over millions of simulated games. We will then compare its performance to the other agent that will use our original strategy that seemed logical to us in social games.

2.0 Literature Review

The Resistance is a hidden role board game of imperfect information where one team's goal is deception and the other team's goal is deduction. Some are given the roles of spies and others the role of resistance. The spies begin the game knowing the roles of everyone and aim to keep their identity as a spy hidden from resistance members. The resistance members begin the game not knowing who is on their team and are given the challenge of working out who the spies are in order to make decisions to win the game. The game is played up to a maximum of five missions. The resistance win by succeeding in three missions, while the spies win by causing three missions to fail.

The Resistance is inherently a game of imperfect information, with potentially hundreds of different possibilities to consider. It is a very rare scenario that a resistance member can have definitive knowledge of even one of the spy members identities. Due to this limited information, it will be useful to leverage inductive logic – a system of logic that extends deduction to less-than-certain inferences (Oaksford & Chater, 2009). Using pure deductive logic (where values are either true or false) would mean ignoring scenarios with less than perfect certainty. This would mean ignoring almost all useful sources of information that an agent can gather and use in a game of *The Resistance*.

Bayesian statistical methods could be of greater help in dealing with uncertainty and conditional probability. Bayes' theorem describes the probability of an event, based on prior knowledge of conditions that might be related to the event. It is of great value in calculating conditional probabilities, because inverse probabilities are typically easier to ascertain than direct probabilities (James 2021). It can describe the relationship between two conditional probabilities, P(A|B) and P(B|A) using the following formula:

$$P(A|B) = \frac{P(B|A)\,P(A)}{\mathrm{P(B)}}$$

Bayes theorem can be applied to *The Resistance* to calculate the probability of each possible configuration of spies given a set of observable actions. For a resistance member, these probabilities can be used to identify players that are more likely to be a spy due to their observed suspicious behaviour. Resistance members can make sure to avoid suspicious missions. On the other hand, spies can use these probabilities to disguise themselves more effectively, choosing resistance-like behaviour wherever possible. In order to implement Bayes theorem, an agent need only work out or estimate the probability of an event occurring given a particular world. Then, it can maintain some sense of which worlds are more or less likely to be true.

Some potential methods of gameplay optimisation include analytical methods, various types of hill-climbing, randomized search, and genetic algorithms. Genetic algorithms are inspired by the process of Darwinian evolution where selection, mutation and crossover play a major role. Good solutions are selected and manipulated to achieve new, and possibly better solutions. The changes in the population are done by genetic operations that work on the "chromosomes" in which the parameters of possible solutions are encoded. In each generation of the genetic algorithm, the new solutions replace the old solutions in the population of solutions (Chapman & Hall, 2000).

A genetic algorithm could be a good choice to further optimise our agents' parameters of strategy. Parameters (or chromosomes) could include opponent model values as well as probabilities for the agent to perform an action when uncertainty is involved. Once these parameters are optimised, Bayes theorem can be implemented with higher accuracy, and our agents actions become more informed and sensible. Appropriate or inappropriate values of parameters will cause the system to perform better or worse, as measured by some relevant objective or fitness function (Chapman & Hall, 2000). Solutions from the population that perform well in this fitness function can then be selected for genetic operators: mutation and crossover functions. Mutation functions alter the genes of a solution while crossover takes parts from two or more solutions and combines them to create a new solution. Zbigniew Michalewicz (1994) demonstrates these genetic operations with the example of a *chromosome* being a string of bits, and each bit being a *gene*. A mutation alters one or more genes by flipping that bit with a probability equal to the chosen mutation rate. The crossover function on the other hand involves choosing two chromosomes and randomly selecting a gene to be the cut-off point. At this cut-off point the values to the left of it are swapped with the values to the left from the other chromosome and vice versa. For example, if the cut-off point was at the fifth gene:

Chromosome1 = (00000|01110000001111)

Chromosome2 = (11100|11001100000000)

Offspring1 = (00000|11001100000000)

Offspring2 = (11100|01110000001111)

Over the course of many iterations of performing the fitness function and genetic operations on successful solutions, the values changed by the genetic algorithm will move towards an optimum.

3.0 Selected Technique

As *The Resistance* involves a lot of uncertainty, we have decided to work with the probabilities of many different possibilities rather than trying to definitively determine the roles of other agents. We will implement this by creating a set of all possible worlds, represented as a list of spies, initially with equal probabilities. We will keep our own agent in the possible worlds even if they are not a spy in order to keep an idea of the suspicions other players might have against our agent. While not giving much certainty, using probabilities would give our agent the best chance of choosing a good mission when a resistance member. Having the most accurate probabilities of these worlds is very useful for the resistance in order to keep spies out of proposed missions. However, it is also valuable for spies to have an idea of how they may be perceived by other agents and adjust their behaviour to look more like a resistance member. Choosing a mission that would most likely pass the voting stage while still including a spy is a key to their success.

The technique selected to update these world probabilities was Bayes' theorem. This will be very useful in a game with so much uncertainty as it allows the agent to convert between different conditional probabilities. The agent will maintain the probability of each world being true, which is updated every round based on three main areas: how players vote for a proposed mission, who proposed that mission, and, most importantly, the outcome of that mission. Our agent will maintain a threshold value for how likely a mission needs to be to succeed in order to vote positively for that mission. This probability of a team's success will also be used when our agents are the team leader. As a resistance member they will simply propose the mission that is most likely to succeed. As a spy they will propose the mission that is most likely to succeed, given that it also includes a spy (always itself unless two fails are needed) in order to have the highest chance of the proposal getting approved.

While we did elect to use inductive probabilities as our primary method of finding beneficial moves, there is still a lot of benefit in using deductive reasoning when possible. We will keep track of impossible worlds by deducing groups of players that could not logically all be spies based on mission outcomes. Sometimes, mission outcomes can even provide concrete evidence that one world is a certainty.

For our second agent we decided to implement a genetic algorithm to find an optimised way for the agent to play. Both of our agents store a range of probabilities of spies and resistance members performing certain actions given a game state, as well as thresholds of how likely a mission needs to be to succeed in order to vote for it. These are the values that will be optimised over time, starting from a base of our best estimates. We will have a population of eight agents at any given time, a fitness function to test how successful agents are, a mutation function to slightly change successful agents and a reproduction function to get an average of successful agents. We will run this algorithm for as long as possible and compare its success to our first agent with hard-coded probabilities.

4.0 Implementation

4.1 Hard coded logical rules

There are some decisions that have will always have one correct action. Despite our goal of determining whether AI can discover winning strategies on its own, it is illogical to remove these hard-coded rules for what will always be the best move. Both agents will always include themselves in a proposed mission, as this is the most likely mission to succeed from a

resistance perspective, and the least suspicious from a spy's perspective. Agents will always vote yes on the fifth mission proposal of the round. Spies never hold the majority and so should never attempt to downvote it. Spies will always betray if it means they win the game, but never betray if there are not enough spies on the mission to fail it. Agents will always upvote the first mission for the sake of progressing the game. All other decisions require more thought and analysis.

4.2 Bayesian Probability

Our Bayesian agent was built over four iterations.

Firstly, we created a Baseline agent that simply obeys our hard-coded logical rules and behaves randomly otherwise.

Next, a set of all worlds and associated probabilities was maintained and updated at the end of each mission. We estimated a probability that a spy would betray a mission for each round and used these values to calculate the probability of a mission outcome given a world.

Let m = mission outcome, w_i = a world configuration, s = number of spies in mission, b = number of betrayals in mission and r = betray rate of a spy in that round. The probability of a mission outcome given a world is given by:

$$P(m \mid w_i) = C_b^s \times r^b \times (1-r)^{s-b}$$

The probability of a mission outcome is the weighted sum of these probabilities for each world by the probability of the world:

$$P(m) = \sum_i P(m \mid w_i) \times P(w_i)$$

Each world probability can be updated by:

$$P(w_i) = P(w_i \mid m) = P(m \mid w_i) \times P(w_i) \div P(m)$$

In our third iteration we estimated the probability that spies and resistance members vote for failed and successful missions in each round. Since we know how each player voted, and we know who the spies and resistance members are in each world, this probability is simply the product of the fixed probabilities of each player voting the way that they did.

Our final iteration resulted in the agent Bayes3, which also updates world probabilities based on our estimated probability of spies and resistance members proposing failed and successful missions. In updating world probabilities, mission outcomes were given a weighting of 1.0, while voting configurations and proposers were given weightings of 0.4 and 0.3 with respect to the previous world probabilities. These numbers were also our hard-coded best estimates of how important each piece of information is.

Next came the challenge of how to use this world information. We can obtain the probability that a player is a spy (their suspicion value) by taking the sum of the probabilities of all worlds in which that player is a spy. The suspicion of a mission is the sum of player suspicions for players in the mission. Our agent votes for a mission only if the mission suspicion is low enough. It also strives to propose a team with the least mission suspicion.

4.3 Genetic Algorithm

At this point in the project, we had completed our first agent Bayes3. The idea behind the implementation of our second agent, Evolver, was to remove all hard-coded behavioural traits and instead create a json file to store mutable data to be passed to the agent. Alterable traits include betray rates, probabilities of spies and resistance members voting and proposing for failed and successful missions, and suspicion thresholds for voting for a mission. In total, the json file stores thirteen traits, each comprised of four float values (a, b, c, d), which combine to form the value of that trait at a particular stage of the game by:

$$T = a \times rounds\ completed^2 + b \times rounds\ completed + c \times failed\ missions + d$$

In order to find optimal opponent modelling values, the Bayesian weightings for voting pattern and proposers were kept constant at 0.4 and 0.3, the same as in Bayes3. It is possible that our genetic agent could have progressed further if these values were freed to evolve once the other parameters displayed some convergence.

7500 games were played per trial. The win rate of each agent was recorded as the fitness function. Each game had a random number of players picked from the pool of the eight evolving agents, plus a random agent, a baseline agent and Bayes3 for control. After a trial, only the top three performing evolving agents survive, the fourth becomes an average of the best and second best, and the fifth becomes an average of the best and the third best. The next two agents become mutations of the best and the final agent becomes a mutation of the second best. The mutation function involves a 50% chance of choosing each behavioural trait and a 50% chance of choosing each a, b, c, or d value to either increase or decrease within that trait. We completed 900 trials over a few days, equating to 6,750,000 games of *The Resistance*.

5.0 Validation of Agent Performance

Throughout development we kept track of each addition of a new source of game information used by the agent. This started by creating an agent that updated world probabilities solely through the outcomes of each mission and the players involved, Bayes. The next addition, Bayes2, looked at how players voted for mission proposals and updated the world probabilities based on how well the voting patterns aligned with the likely voting patterns of each world. Finally, in Bayes3, we considered the proposer of each mission, looking at who they wanted in the mission and again updated the word probabilities based on how well the proposal aligned with what was likely for each world. This had left us with three main versions of the first agent. The hope was that with each source of information added the performance would improve. For comparison we also created a baseline agent Baseline that did logical moves when possible, but did not keep track of the probabilities of worlds. To test the effectiveness of each addition we ran 100,000 simulated games, the number of players in each game being a random selection of five up to ten, and each player was randomly selected from our pool of agents. The results were as follows.

Agent	Spy win rate	Resistance win rate	Overall win rate
Random	0.39185	0.41719	0.4076
Baseline	0.56952	0.42037	0.47716
Bayes	0.57886	0.466	0.50849
Bayes2	0.58585	0.4826	0.52134
Bayes3	0.61839	0.48887	0.53775

As seen from the results, developing our first agent went as we had hoped and expected. Keeping track of all spy team possibilities and updating them based on the outcome of missions made a significant difference especially when playing as a resistance member as can be seen from the difference between Baseline and Bayes. Considering who proposed and who voted for individual rounds was also evidently useful information as Bayes3 outperformed all others as both a spy and a resistance member.

During the process of running the genetic algorithm for our second agent, we saved the genes generated after 250, 700 and 900 trials. We played another 100,000 random games with the agents below:

Agent	**Spy win rate**	**Resistance win rate**	**Overall win rate**
Random	0.41923	0.30632	0.349
Baseline	0.62407	0.29919	0.42152
Bayes3	0.64344	0.36286	0.4687
Evolver250	0.71154	0.40034	0.51763
Evolver700	0.7162	0.41292	0.52735
Evolver900	0.72147	0.41651	0.53256

And finally, we also ran a direct comparison between our final two agents Bayes3 and Evolved (Evolved having the same genes as Evolver900):

Agent	**Spy win rate**	**Resistance win rate**	**Overall win rate**
Bayes3	0.59415	0.32372	0.42586
Evolved	0.69543	0.39409	0.50795

Our genetic algorithm was able to iteratively improve the values used in both the Bayesian updating and the decision-making using Bayesian probabilities, proving that artificial intelligence is more effective at optimising a Bayesian agent than human estimations. Our final evolved agent improved upon Bayes3's overall win rate by 13.6% in the tournament with multiple agents and by 8.2% when they are the only agents in the tournament. It would be interesting to witness the performance of Evolver after Bayesian weightings are freed to evolve, and more behavioural traits and state information is included in the mutable data.

References

Joyce, James, "Bayes' Theorem", *The Stanford Encyclopaedia of Philosophy* (Fall 2021 Edition), Edward N. Zalta (ed.), URL - <https://plato.stanford.edu/archives/fall2021/entries/bayes-theorem/>.

Oaksford, M, & Chater, N. (2009). The uncertain reasoner: Bayes, logic, and rationality. Behavioral and Brain Sciences, 32(1). 105-120. Retrieved from https://doi-org.ezproxy.library.uwa.edu.au/10.1017/S0140525X0900051X

Chapman, H. C. (2000). *The Practical Handbook of Genetic Algorithms* (L. D. Chambers, Ed. Vol. 2). https://www.taylorfrancis.com/books/mono/10.1201/9781420035568/practical-handbook-genetic-algorithms-lance-chambers

Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs* (2 ed.). https://ebookcentral.proquest.com/lib/UWA/detail.action?docID=3100463&pq-origsite=primo